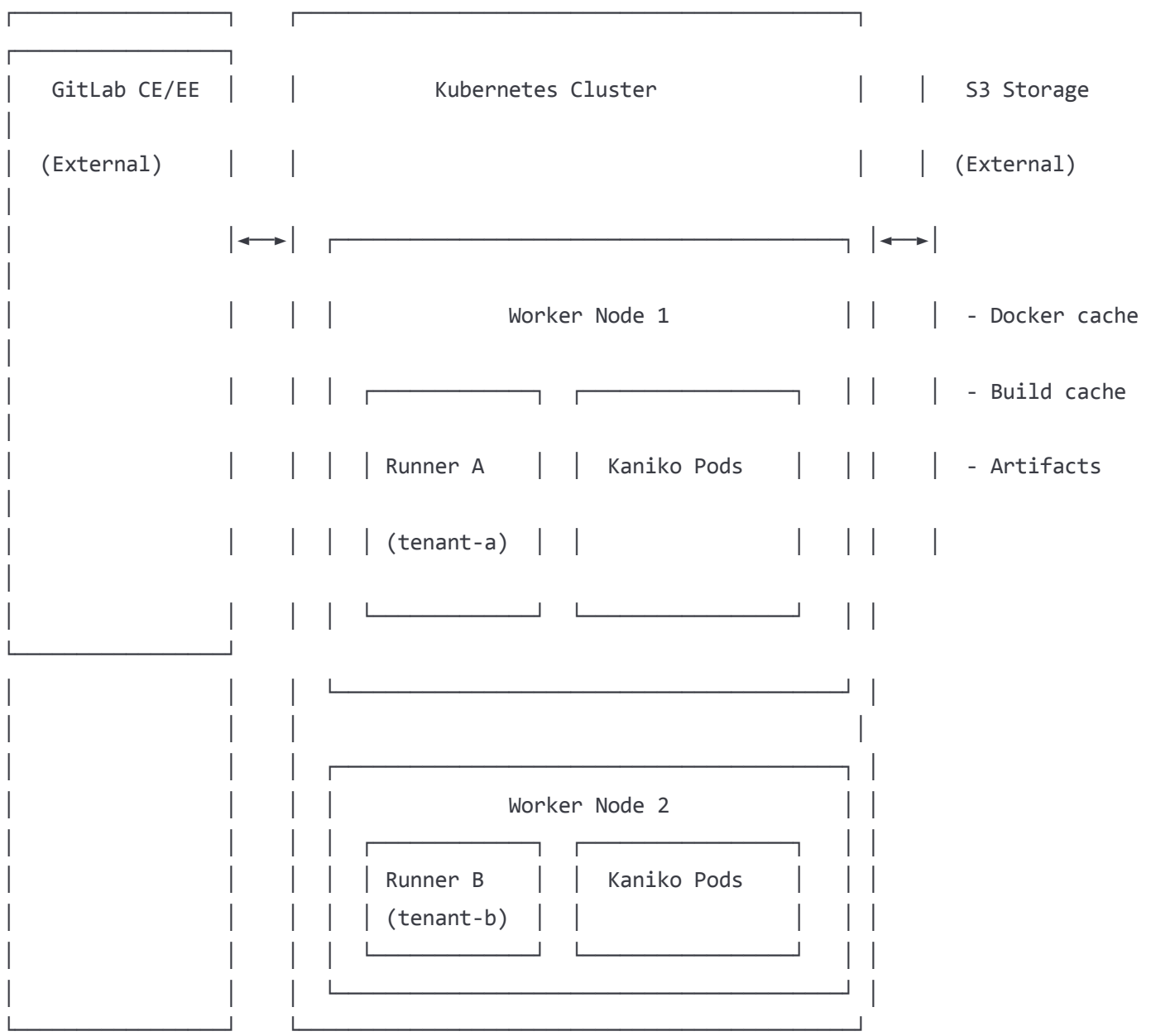
Архитектура GitLab Runners с Kaniko в Kubernetes

Общая архитектура



Ключевые компоненты

1. GitLab Runners

- **Executor**: Kubernetes
- **Мультитенантность**: Отдельные Runners для каждого тенанта
- **Параллелизм**: До 20 заданий на Runner
- **Фиксированные узлы**: nodeSelector/nodeAffinity для привязки к конкретным worker узлам

2. Kaniko

- **Daemonless**: Сборка контейнеров без Docker daemon

- **Multistage builds**: Поддержка сложных Dockerfile
- **Кеширование**: Использование S3 для кеша слоев
- **Изоляция**: Отдельные кеши для каждого тенанта

3. Изоляция и безопасность

- **Namespaces**: Отдельные namespace для каждого тенанта
- **RBAC**: Ограниченные права доступа
- **NetworkPolicies**: Сетевая изоляция между тенантами
- **NodeAffinity**: Привязка к фиксированным узлам

Схема работы

1. **Регистрация Runner**: Каждый Runner регистрируется в GitLab с уникальными тегами
2. **Получение задания**: Runner получает задание из GitLab
3. **Создание Pod**: Runner создает Pod с Kaniko на назначенном узле
4. **Сборка**: Kaniko выполняет multistage сборку с использованием кеша из S3
5. **Публикация**: Готовый образ публикуется в Docker Registry
6. **Очистка**: Pod удаляется после завершения задания

Преимущества архитектуры

- ✅ **Безопасность**: Изоляция на уровне namespace и узлов
- ✅ **Масштабируемость**: Горизонтальное масштабирование Runners
- ✅ **Эффективность**: Кеширование и параллельная обработка
- ✅ **Мультитенантность**: Полная изоляция между тенантами
- ✅ **Надежность**: Отказоустойчивость через Kubernetes

Требования к ресурсам

Worker Node

- **CPU**: Минимум 8 cores на узел
- **RAM**: Минимум 16GB на узел
- **Storage**: SSD для временных файлов сборки
- **Network**: Высокая пропускная способность для загрузки образов

Pod ресурсы (Kaniko)

- **CPU**: 1-2 cores на задание
- **RAM**: 2-4GB на задание
- **Ephemeral Storage**: 10-20GB на задание